Protocollo Diffie-Hellman

Scambio Sicuro di Chiavi Crittografiche

Valerio Vaccaro

Satoshi Spritz Connect

28 Ottobre 2025

Valerio Vaccaro ∞

- 💻 Sviluppatore Bitcoin ed Esperto Hardware
- 🔥 Contributore a progetti Bitcoin open source
- ⚠️ Appassionato di hardware fai-da-te (DIY)
- Ingegnere Bitcoin e Liquid presso Blockstream

Social

- 🧑‍💼 **LinkedIn** linkedin.com/in/valeriovaccaro
- 🐙 **Github** github.com/valerio-vaccaro
- **Telegram** t.me/valeriovaccaro

Licenza

May this presentation inspire you to become more self-sovereign!

Meme

Introduzione al Protocollo Diffie-Hellman

Cos'è il Protocollo Diffie-Hellman?

- 🤝 **Protocollo di scambio chiavi** inventato nel 1976
- 🛡️ Permette a due parti di **generare una chiave segreta condivisa**
- 🔢 **Sicurezza basata sulla difficoltà computazionale** del problema del logaritmo discreto
- 🔒 Fondamentale per la **sicurezza delle comunicazioni moderne**

Punti Chiave

- 💡 Rivoluzionario: prima implementazione pratica di scambio chiavi
- 🔥 Base per molti protocolli di sicurezza (TLS, SSH, VPN)
- ⚠️ Vulnerabile agli attacchi man-in-the-middle senza autenticazione

Il Problema Matematico

Logaritmo Discreto

Dati: - **p**: numero primo grande - **g**: generatore modulo p - **A = g^a mod p**: valore pubblico - **a**: esponente segreto
Problema: trovare **a** conoscendo solo **A**, **g**, **p**

Esempio Pratico

`p = 23, g = 5`
`a = 6 → A = 5^6 mod 23 = 8`
Per trovare **a** da **A = 8**, devi provare tutti i valori possibili!

⚙ Come Funziona il Protocollo

Setup Iniziale

Alice 👩 - Sceglie numero primo **p** - Sceglie generatore **g** - Sceglie segreto **a** - Calcola **A = gˆa mod p**

Bob 👨 - Riceve **p**, **g**, **A** - Sceglie segreto **b** - Calcola **B = gˆb mod p** - Invia **B** ad Alice

Scambio delle Chiavi

1. ✉ Alice invia **(p, g, A)** a Bob
2. ✉ Bob invia **B** ad Alice
3. 🔑 Entrambi calcolano **K = gˆ(ab) mod p**

Esempio Numerico

Parametri

- **p = 23** (numero primo)
- **g = 5** (generatore)
- **a = 6** (segreto di Alice)
- **b = 15** (segreto di Bob)

Calcoli

Alice calcola: - **A = 5^6 mod 23 = 8** - **K = 19^6 mod 23 = 2**

Bob calcola: - **B = 5^15 mod 23 = 19** - **K = 8^15 mod 23 = 2**

Risultato

Chiave condivisa K = 2

Proprietà di Sicurezza

Cosa è Protetto

- 🔒 **I segreti a e b** non vengono mai trasmessi
- 👀 **La chiave finale K** non viaggia sulla rete
- 🔥 **Anche intercettando tutto il traffico**, l'attaccante non può ricavare K

Cosa NON è Protetto

- ⚠️ **Attacchi Man-in-the-Middle**: senza autenticazione, un attaccante può impersonare entrambe le parti
- 💀 **Attacchi quantistici**: i computer quantistici potrebbero (in futuro) rompere il logaritmo discreto

Implementazioni Pratiche

Varianti Moderne

ECDH ⚙ - Curve ellittiche invece di numeri primi - Chiavi più piccole - Usato in Bitcoin, TLS, SSH

X25519 🚀 - Curve di Montgomery - Molto veloce - Standard moderno

Uso in Bitcoin

- 🪙 **Scambio di chiavi** per comunicazioni sicure
- 🔒 **Protocolli di comunicazione** tra wallet
- 🛡️ **Autenticazione** in alcuni protocolli Lightning

⚠ Vulnerabilità e Attacchi

Man-in-the-Middle Attack

`Alice ↤↦ Attacker ↤↦ Bob`

1. 👀 Attaccante intercetta comunicazioni
2. 😈 Impersona Bob con Alice e Alice con Bob
3. 🔑 Stabilisce chiavi separate con entrambi
4. 💀 Decifra tutto il traffico

Contromisure

- 📜 **Autenticazione delle chiavi pubbliche**
- 🤝 **Certificati digitali**
- 🔒 **Out-of-band verification**

Evoluzione e Standard

Timeline Storica

- **1976**: Whitfield Diffie e Martin Hellman pubblicano il protocollo
- **1985**: Primi standard governativi (NSA Suite A)
- **1990s**: Diffusione commerciale (SSL/TLS)
- **2000s**: Curve ellittiche (ECDH)
- **2010s**: X25519 e ChaCha20-Poly1305

Standard Moderni

- **RFC 7748**: X25519 e X448
- **RFC 8446**: TLS 1.3 con ECDH
- **NIST SP 800-56A**: Standard governativi

Implementazione Pratica

Python Example

```python
import wallycore as wally

# Alice genera coppia di chiavi
alice_private = bytes(os.urandom(32))
alice_public = wally.ec_public_key_from_private_key(alice_private)

# Bob genera coppia di chiavi
bob_private = bytes(os.urandom(32))
bob_public = wally.ec_public_key_from_private_key(bob_private)
```

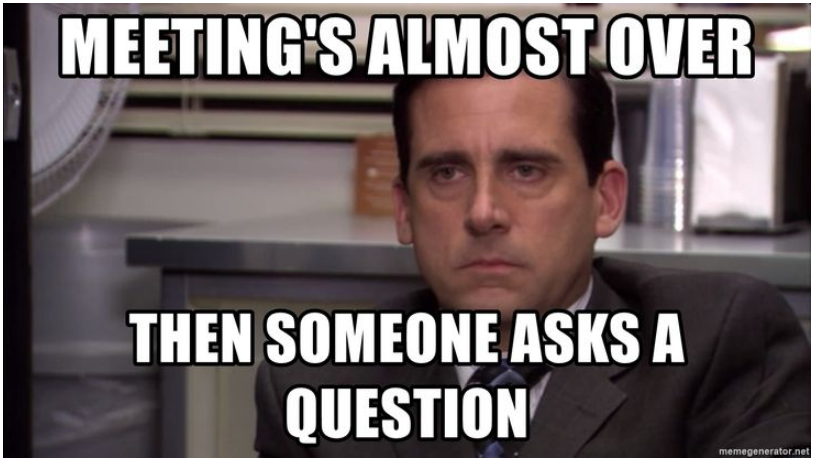
Implementazione Pratica (continua)

Python Example

```
# Calcolo chiavi
alice_shared = wally.ecdh(bob_public, alice_private)
bob_shared = wally.ecdh(alice_public, bob_private)

# alice_shared == bob_shared
```

? Domande


MEETING'S ALMOST OVER
THEN SOMEONE ASKS A QUESTION
memegenerator.net

Bibliografia

- Whitfield Diffie, Martin Hellman. “New Directions in Cryptography” (1976)
- RFC 7748: “Elliptic Curves for Security” (2016)
- RFC 8446: “The Transport Layer Security (TLS) Protocol Version 1.3” (2018)
- NIST SP 800-56A: “Recommendation for Pair-Wise Key Establishment Schemes” (2018)
- Daniel J. Bernstein. “Curve25519: new Diffie-Hellman speed records” (2006)

Progetto Satoshi Spritz

- 💻 Federazione di gruppi locali di Bitcoiner
- 🎓 Eventi gratuiti e privacy oriented
- 🤖 BITCOIN ONLY
- 🔧 Satoshi Spritz Connect online settimanale
- 📖 Orientato all'apprendimento della self-sovereign
- 🍕 Tutte le settimane un evento online -> Satoshi Spritz Connect

Links

- satoshispritz.it
- t.me/SatoshiSpritzConnect

Officine Bitcoin

- Comunità Italiana di Bitcoiners, totalmente gratuita
- BITCOIN ONLY
- Focus su educazione e sviluppo di progetti
- Progetti:
  - Sviluppo nodi Bitcoin
  - Uso di Hardware Wallet
  - Filosofia open source
  - Installazione di Debian
  - Mnemoniche & Dadi
  - . . . e molto altro

Links

- officinebitcoin.it